Terms used w. r. stevens tcp/ip illustrated vol. 1 Found 86,771 of 132,857

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 The transport layer: tutorial and survey

Sami Iren, Paul D. Amer, Phillip T. Conrad

December 1999 **ACM Computing Surveys (CSUR)**, Volume 31 Issue 4

Full text available: pdf(261.78 KB) Additional Information: full citation, abstract, references, citings, index terms

Transport layer protocols provide for end-to-end communication between two or more hosts. This paper presents a tutorial on transport layer concepts and terminology, and a survey of transport layer services and protocols. The transport layer protocol TCP is used as a reference point, and compared and contrasted with nineteen other protocols designed over the past two decades. The service and protocol features of twelve of the most important protocols are summarized in both text and tables.< ...

Keywords: TCP/IP networks, congestion control, flow control, transport protocol, transport service

2 Column: A case for context-aware TCP/IP

Carey Williamson, Qian Wu

March 2002 **ACM SIGMETRICS Performance Evaluation Review**, Volume 29 Issue 4

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references

This paper discusses the design and evaluation of CATNIP, a Context-Aware Transport/Network Internet Protocol for the Web. This integrated protocol uses application-layer knowledge (i.e., Web document size) to provide explicit context information to the TCP and IP protocols. While this approach violates the traditional layered Internet protocol architecture, it enables informed decision-making, both at network endpoints and at network touters, regarding flow control, congestion control, and pack ...

Keywords: TCP/IP, internet protocols, network emulation, network simulation, web performance

3 Using certes to infer client response time at the web server

David Olshefski, Jason Nieh, Dakshi Agrawal

February 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 1

Full text available: pdf(2.30 MB) Additional Information: full citation, abstract, references, index terms

As businesses continue to grow their World Wide Web presence, it is becoming increasingly vital for them to have quantitative measures of the mean client perceived response times of their web services. We present Certes (CliEnt Response Time Estimated by the Server), an online server-based mechanism that allows web servers to estimate mean client perceived response time, as if measured at the client. Certes is based on a model of TCP that quantifies the effect that connection drops have on mean ...

Keywords: Web server, client perceived response time

4 Improving TCP performance over mobile networks

Hala Elaarag
September 2002 **ACM Computing Surveys (CSUR)**, Volume 34 Issue 3

Full text available: pdf(219.39 KB) Additional Information: full citation, abstract, references, citings, index terms

Transmission Control Protocol (TCP) is the most commonly used transport protocol on the Internet. All indications assure that mobile computers and their wireless communication links will be an integral part of the future internetworks. In this paper, we present how regular TCP is well tuned to react to packet loss in wired networks. We then define mobility and the problems associated with it. We discuss why regular TCP is not suitable for mobile hosts and their wireless links by p ...

Keywords: I-TCP, M-TCP, MTCP, New-Reno, Reno, SACK, TCP performance, WAP, WTCP, base station, comparison of TCP implementations, end-to-end, link layer, mobile TCP, mobile host, mobile wireless networks, mobility, snoop, split TCP, standard TCP, wired networks, wireless TCP

5 The state of the art in locally distributed Web-server systems

Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, Philip S. Yu
June 2002 **ACM Computing Surveys (CSUR)**, Volume 34 Issue 2

Full text available: pdf(1.41 MB) Additional Information: full citation, abstract, references, citings, index terms

The overall increase in traffic on the World Wide Web is augmenting user-perceived response times from popular Web sites, especially in conjunction with special events. System platforms that do not replicate information content cannot provide the needed scalability to handle large traffic volumes and to match rapid and dramatic changes in the number of clients. The need to improve the performance of Web-based services has produced a variety of novel content delivery architectures. This article w ...

Keywords: Client/server, World Wide Web, cluster-based architectures, dispatching algorithms, distributed systems, load balancing, routing mechanisms

6 Timescales and stability: A non-instrusive, wavelet-based approach to detecting network performance problems

Polly Huang, Anja Feldmann, Walter Willinger
November 2001 **Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement**

Full text available: pdf(3.01 MB) Additional Information: full citation, abstract, references, citings, index terms

The main objective of this paper is to explore how much information about the characteristics of end-to-end network paths can be inferred from relying solely on passive packet-level traces of existing traffic collected from a single tap point in the network. To this

end, we show that a number of structural properties of aggregate TCP/IP packet traces reveal themselves and can be compared across different time periods and across paths of the traffic destined to different subnets by exploiting the ...

Keywords: energy function, network performance, passive measurements, scale-localization, wavelets

7 Fast address lookups using controlled prefix expansion
V. Srinivasan, G. Varghese
February 1999 **ACM Transactions on Computer Systems (TOCS)**, Volume 17 Issue 1

Full text available: pdf(258.60 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Internet (IP) address lookup is a major bottleneck in high-performance routers. IP address lookup is challenging because it requires a longest matching prefix lookup. It is compounded by increasing routing table sizes, increased traffic, higher-speed links, and the migration to 128-bit IPv6 addresses. We describe how IP lookups and updates can be made faster using a set of of transformation techniques. Our main technique, controlled prefix expansion, transf ...

Keywords: Internet address lookup, binary search on levels, controlled prefix expansion, expanded tries, longest-prefix match, multibit tries, router preformance

8 A methodology for analyzing the performance of authentication protocols
Alan Harbitter, Daniel A. Menascé
November 2002 **ACM Transactions on Information and System Security (TISSEC)**, Volume 5 Issue 4

Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, references, index terms

Performance, in terms of user response time and the consumption of processing and communications resources, is an important factor to be considered when designing authentication protocols. The mix of public key and secret key encryption algorithms typically included in these protocols makes it difficult to model performance using conventional analytical methods. In this article, we develop a validated modeling methodology to be used for analyzing authentication protocol features, and we use two ...

Keywords: Authentication, Kerberos, mobile computing, performance modeling, proxy servers, public key cryptography

9 How a large ATM MTU causes deadlocks in TCP data transfers
Kjersti Moldeklev, Per Gunningberg
August 1995 **IEEE/ACM Transactions on Networking (TON)**, Volume 3 Issue 4

Full text available: pdf(1.43 MB) Additional Information: full citation, references, citings, index terms

10 Comparative study of various TCP versions over a wireless link with correlated losses
Farooq Anjum, Leandros Tassiulas
June 2003 **IEEE/ACM Transactions on Networking (TON)**, Volume 11 Issue 3

Full text available: pdf(737.09 KB) Additional Information: full citation, abstract, references, index terms

In this paper, we investigate the behavior of the various Transmission Control Protocol (TCP) algorithms over wireless links with correlated packet losses. For such a scenario, we show that the performance of NewReno is worse than the performance of Tahoe in many

situations and even OldTahoe in a few situations because of the inefficient fast recovery method of NewReno. We also show that random loss leads to significant throughput deterioration when either the product of the square of the bandwi ...

Keywords: TCP algorithm, TCP over wireless, correlated losses, packet train model, performance analysis

11 Research contributions: Facilitating localized exploitation and enterprise-wide integration in the use of IT infrastructures: the role of PC/LAN infrastructure standards
Timothy R. Kayworth, V. Sambamurthy
September 2000 **ACM SIGMIS Database**, Volume 31 Issue 4

Full text available: pdf(2.61 MB) Additional Information: full citation, abstract, references, index terms

Two often-contradictory dilemmas confront firms in their efforts at promoting IT-based innovation: facilitate localized exploitation of the IT infrastructure within individual business units, but also ensure enterprise-wide integration of IT innovation initiatives to facilitate business applications that exploit inter-unit synergies. Therefore, developing an IT infrastructure that is conducive to both localized exploitation and enterprise-wide integration is important. In this context, IT infras ...

Keywords: IS implementation, case study, management information systems, organizational structure, strategic issues

12 Wide area traffic: the failure of Poisson modeling
Vern Paxson, Sally Floyd
June 1995 **IEEE/ACM Transactions on Networking (TON)**, Volume 3 Issue 3

Full text available: pdf(2.18 MB) Additional Information: full citation, references, citings, index terms

13 TCP over ATM: ABR or UBR?
Teunis J. Ott, Neil Aggarwal
June 1997 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 25 Issue 1

Full text available: pdf(1.74 MB) Additional Information: full citation, abstract, references, index terms

This paper reports on a simulation study of the relative performances of the ATM ABR and UBR service categories in transporting TCP/IP flows through an ATM Network. The objective is two-fold: (i) to understand the interaction between the window - based end-to-end flowcontrol TCP and the rate based flowcontrol ABR which is restricted to the ATM part of the network, and (ii) to decide whether the greater complexity of ABR (than UBR) pays off in better performance of ABR (than UBR).The most importa ...

14 Parallel shared-memory simulator performance for large ATM networks
Brian Unger, Zhonge Xiao, John Cleary, Jya-Jang Tsai, Carey Williamson
October 2000 **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, Volume 10 Issue 4

Full text available: pdf(223.11 KB) Additional Information: full citation, abstract, references, citings, index terms, review

A performance comparison between an optimistic and a conservative parallel simulation kernel is presented. Performance of the parallel kernels is also compared to a central-event-list sequential kernel. A spectrum of ATM network and traffic scenarios representative of those used by ATM networking researchers are used for the comparison. Experiments are

conducted with a cell-level ATM network simulator and an 18-processor SGI PowerChallenge shared-memory multiprocessor. The resul ...

Keywords: ATM network modeling, conservative synchronization, optimistic synchronization, parallel discrete event simulation, time warp

15 Real-time estimation of the parameters of long-range dependence
Matthew Roughan, Darryl Veitch, Patrice Abry
August 2000 **IEEE/ACM Transactions on Networking (TON)**, Volume 8 Issue 4
Full text available: pdf(237.43 KB) Additional Information: full citation, references, citings, index terms

Keywords: Hurst parameter, estimation, fractal, long-range dependence, on-line, real-time, self-similar, traffic modeling, wavelets

16 On achievable service differentiation with token bucket marking for TCP
Sambit Sahu, Philippe Nain, Christophe Diot, Victor Firoiu, Don Towsley, Don Iowsley
June 2000 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 28 Issue 1
Full text available: pdf(876.53 KB) Additional Information: full citation, abstract, references, citings, index terms

The Differentiated services (diffserv) architecture has been proposed as a scalable solution for providing service differentiation among flows without any per-flow buffer management inside the core of the network. It has been advocated that it is feasible to provide service differentiation among a set of flows by choosing an appropriate "marking profile" for each flow. In this paper, we examine (i) whether it is possible to provide service differentiation among a set of TCP flow ...

17 Building real-time groupware with GroupKit, a groupware toolkit
Mark Roseman, Saul Greenberg
March 1996 **ACM Transactions on Computer-Human Interaction (TOCHI)**, Volume 3 Issue 1
Full text available: pdf(2.74 MB) Additional Information: full citation, abstract, references, citings, index terms, review

This article presents an overview of GroupKit, a groupware toolkit that lets developers build applications for synchronous and distributed computer-based conferencing. GroupKit was constructed from our belief that programming groupware should be only slightly harder than building functionally similar single-user systems. We have been able to significantly reduce the implementation complexity of groupware through the key features that comprise GroupKit. A runtime infrastructure
Keywords: GroupKit, computer-supported cooperative work, groupware toolkits, synchronous groupware, user interface toolkits

18 Cost profile of a highly assured, secure operating system
Richard E. Smith
February 2001 **ACM Transactions on Information and System Security (TISSEC)**, Volume 4 Issue 1
Full text available: pdf(165.98 KB) Additional Information: full citation, abstract, references, citings, index terms

The Logical Coprocessing Kernel (LOCK) began as a research project to stretch the state of

the art in secure computing by trying to meet or even exceed the "A1" requirements of the Trusted Computer System Evaluation Criteria (TCSEC). Over the span os seven years, the project was transformed into an effort to develop and deploy a product: the Standard Mail Guard (SMG). Since the project took place under a US government contract, the development team needed to maintain detailed re ...

Keywords: LOCK (Logical Coprocessing Kernel), security kernels

19 Understanding TCP Vegas: a duality model

Steven H. Low, Larry L. Peterson, Limin Wang

March 2002 **Journal of the ACM (JACM)**, Volume 49 Issue 2

Full text available: pdf(437.98 KB) Additional Information: full citation, abstract, references, citings, index terms

We view congestion control as a distributed primal--dual algorithm carried out by sources and links over a network to solve a global optimization problem. We describe a multilink multisource model of the TCP Vegas congestion control mechanism. The model provides a fundamental understanding of delay, fairness and loss properties of TCP Vegas. It implies that Vegas stabilizes around a weighted proportionally fair allocation of network capacity when there is sufficient buffering in the network. It ...

Keywords: Persistent congestion, REM, TCP Vegas, TCP congestion control

20 Efficient user-space protocol implementations with QoS guarantees using real-time upcalls

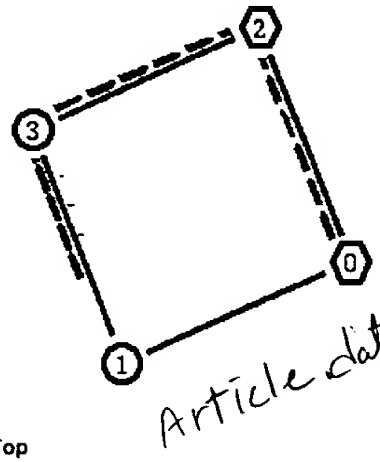
R. Gopalakrishnan, Gurudatta M. Parulkar

August 1998 **IEEE/ACM Transactions on Networking (TON)**, Volume 6 Issue 4

Full text available: pdf(205.42 KB) Additional Information: full citation, references, citings, index terms

Keywords: multimedia communication, networks, operating system kernals, processor scheduling, protocols, real-time systems, transport protocols

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next



NIUNet

"Now I Understand Networking!"

Understanding Networking Principles through Visualization, Simulation, Emulation, and Application

Visualization:
- TCP Visualizations
- Java Visualizer

Simulation:

Emulation:

Application:
- Hacking
- Example Applications
- Packet Traces
- HackNet

Related Work:
- Internet Resources

Smurf Attack

About This Page

This page was researched and constructed by Jonathan Duty (undergraduate) and Sunitha Ramakrishna (graduate). Its main purposes are to explain the concept of Smurf Attacks and hopefully present some ideas on how to prevent them.

What is a Smurf Attack?

A Smurf Attack is a denial-of-service network attack (DoS) that is directed towards some pre-determined target, usually a server. Usually the targets for these attacks are IRC servers, but any server that is plugged into a network and can receive IP packets is vulnerable. These attacks come very quickly and present themselves as very hard to trace.

One of the most visual uses of a Smurf Attack was when the Yahoo server was taken down for 3 hours by such an attack. Once the server was being flooded with **request** packets, others who wanted to connect to it were not able. This may not seem like a big deal, but then take into consideration how much money the

Yahoo web site makes in an hour, multiply that figure by three, and that is how much money they lost due to this crime.

Performing a Smurf Attack involves creating an ICMP **packet**, usually an **echo** or a ping **request packet**, and placing the victim's address in the return field, thus forging the **packet**. This **packet** is then broadcasted onto the network, being received by several hosts who blindly reply to the victim with a response. The victim, now receiving several times its usual load, is overwhelmed with response packets.

Who is Involved in a Smurf Attack?

The Attacker

A perpetrator who selects a victim and forges **request** packets. It then broadcasts them onto the network in hope of harming the victim in some form or fashion.

The Victim

The unassuming machine who is the target selected by the attacker, and whose IP address is spoofed.

Intermediaries

This title can be applied to a number of different roles in a Smurf Attack. Basically, an intermediary is someone who unknowingly harms the victim due to the trickery of the attacker. These people are:

- The Amplifiers: These are the routers and other machines on the Internet that help the attacker broadcast its spoofed **echo request packet** to as many hosts as possible.
- The Hosts: These are the machines that received the spoofed packets off the network and unknowingly reply to the victim.

How to Perform a Smurf Attack

Here is a step-by-step summary to perform a Smurf Attack:

1. Pick a victim. The victim needs to be a machine which will be able to receive the **echo** response packets.
2. Create a ICMP **echo request packet** and address it as broadcast.
3. Place the victim's IP address into return field (sin_addr).
4. Place the **packet** onto the network to be broadcasted.
5. Watch the destruction!!

The Code We Used

This is the code we used to run our experiment. PLEASE DO NOT USE THIS CODE FOR DESTRUCTION. This is only for educational purposes only. It was written by a person who calls him- or herself TFreak and it can be found at http://rootshell.com/archive-j457nxiqi3gq59dv/199710/smurf.c.html or locally

Some Ideas on Preventing a Smurf Attack

One of the reasons the Smurf Attack is so ideal for the perpetrator is that it is extremely difficult to prevent without sacrificing other system functions. Since the Smurf Attack needs a set of players to ensure its success, each player can play a role in preventing it. Several methods are particular to an individual machine or software, but there do exist some generic precautions. We will individually present tactics these players can use in prevention.

Routers:

Since routers are usually what are used to broadcast the spoofed packets onto the network, there are some things that can be done to help cripple this attack:

Router Interface Checking is one way is to check the return address (which is the part of the **packet** that is spoofed) with the routing table whenever a **packet** crosses a router. If the return address is coming through an interface on the router that doesn't match up, then there is obviously something wrong. This has a few problems. This only works if the perpetrator and the victim are on different interfaces on the router. If they are on the same interface, the router will have no way of knowing which machine it came from.

Disabling Broadcast Capabilities may be an option offered by some routers. Select routers (based on their vendor and model) have the capability to turn off the ability to receive network-prefix-directed broadcasts on one of its interfaces. All routers MUST have the capability to disable forwarding network-prefix-directed broadcasts. These two capabilities prevents the router from forwarding broadcast packets from one interface to another. *Check with your individual vendor on ways to prevent your router from forwarding or receiving network-prefixed-directed broadcasts.

The routers can also **sniff IP packets designated for your high-profile machines** on your network. Basically, the routers must have a list of all "high-profile" machines that you have plugged into the network on one of its interfaces. If the router begins to receive several IP **echo** reply packets designated for those machines, they

allow the packets to be dropped at the soonest level. Unfortunately, bandwidth is still used up by the attack until the point at the router when they are dropped and others lose the ability to receive **echo** replies from the high-profile machines on the network.

Hosts can be set up not to respond to ICMP **echo request** packets. This is usually through a patch, and even though this can be done easily, it takes away from the functionality of the host. This is a very hot debate topic because there are those who believe that the ability to respond to **echo request** packets provides a very useful and important diagnostic tool within a network.

A new technology called **CAR** (committed access rate) has just been introduced by the routing world. It is a piece of software that goes into certain routers that allows administrators to limit what types of traffic can go to different types of hosts. So if you don't want **echo** reply packets going to your main web server, then you could use this software to disable that event from occurring.

Links and Acknowledgements

This page talks about a defenses against Smurf Attacks that MCI has created:
http://www.internetnews.com/isp-news/1997/10/0901-mci.htm

This site talks about how Yahoo was taken down for 3 hours by a Smurf Attack:
http://www.time.com/time/digital/daily/0,2822,38899,00.html

This is a paper written by Craig A. Hugen about Smurf Attacks:
http://users.quadrunner.com/chuegen/smurf.txt

---

webmonkey/backend/

Examine Your Network with Ping and Traceroute

by Rob Robertson

Web pages can be slow for any number of reasons, but it usually comes down to this: either the network is slow and congested, or the actual server you're accessing is heavily loaded and taking a while to process your request.

It is hard to diagnose a server problem remotely, but some basic tools can help you check out the network. The three tools I use most often to debug networks are called **ping**, **traceroute**, and **telnet**. All three tools originated under Unix, but have spawned DOS and Windows programs that behave similarly (namely ping and **tracert**, which are available using the DOS command shell). There are also versions of these programs that work on a Macintosh. For the purposes of this article, I'm assuming you are using Unix or Linux, though it should apply directly to DOS and Windows.

In this article I am going start with ping, then show how to use Traceroute. What ping does is send out a special packet called the **Internet Control Message Protocol (ICMP) Echo Request packet**. ICMP packets are special IP control messages that are used to send network information between two hosts, usually things like "don't do that," "send fewer packets," "we don't provide what you want," and "don't go here - go there." When a machine receives an Echo Request, it responds with an Echo Reply, placing the original Echo Request packet into the data field of the Echo Reply.

The way you use ping is pretty simple: You enter `ping hostname`, where `hostname` can be a machine name or just an IP address. There are many different versions of ping. The really lame ones just print out, "hostname is alive" (which is only cool if you are pinging a host named elvis). If you have a good version of ping installed, it will produce output like:

```
$ ping www.webmonkey.com
PING webmonkey.com (198.168.37.209): 56 data bytes
64 bytes from 198.168.37.209: icmp_seq=0 ttl=253 time=0.398 ms
64 bytes from 198.168.37.209: icmp_seq=1 ttl=253 time=0.552 ms
64 bytes from 198.168.37.209: icmp_seq=2 ttl=253 time=0.554 ms
64 bytes from 198.168.37.209: icmp_seq=3 ttl=253 time=0.553 ms
64 bytes from 198.168.37.209: icmp_seq=4 ttl=253 time=0.554 ms
64 bytes from 198.168.37.209: icmp_seq=5 ttl=253 time=0.551 ms
64 bytes from 198.168.37.209: icmp_seq=6 ttl=253 time=0.552 ms
64 bytes from 198.168.37.209: icmp_seq=7 ttl=253 time=0.554 ms
64 bytes from 198.168.37.209: icmp_seq=8 ttl=253 time=0.554 ms
```

```
64 bytes from 198.168.37.209: icmp_seq=9 ttl=253 time=0.553 ms
^C
        ----localhost PING Statistics----
        10 packets transmitted, 10 packets received, 0% packet loss
        round-trip min/avg/max = 0.398/0.537/0.554 ms    $
```

What's happening is that I'm pinging www.webmonkey.com - that is, sending one ICMP Echo Request packet, every second, to www.webmonkey.com. When the ping program gets an Echo Reply back from the remote host (www.webmonkey.com), it prints out the response, giving me several interesting pieces of information: the first is the IP address of where it came from (usually this should be the IP address of www.webmonkey.com); the second is the sequence number, starting at 0; the third is the Time To Live field; and the last is the number of milliseconds it took to get a reply.

The sequence number indicates which ping packet it got a reply to. A skipped sequence number indicates a dropped packet, meaning either the Echo Request or the responding Echo Reply somehow got lost in the network. If this is low (low is under 1 percent of the time), there is nothing to worry about. If it is much higher, the network link has some problems. It's not unheard of to get occasional out-of-order responses or multiple replies, but if you get a lot of them, there's trouble.

The Time To Live (TTL) field can be interesting. Every IP packet that gets sent out has a TTL field which is set to a relatively high number (in this case, ping packets get a TTL of 255). As the packet traverses the network, the TTL field gets decreased by one by each router it goes through; when the TTL drops to 0, the packet is discarded by the router. The IP spec says that the TTL should be set to 60 (though it's 255 for ping packets). The main purpose of this is so that a packet doesn't live forever on the network and will eventually die when it is deemed "lost." But for us, it provides additional information. We can use the TTL to determine approximately how many router hops the packet has gone through. In this case it's 255 minus *N* hops, where *N* is the TTL of the returning Echo Replies. If the TTL field varies in successive pings, it could indicate that the successive reply packets are going via different routes, which isn't a great thing.

The time field is an indication of the round-trip time to get a packet to the remote host. The reply is measured in milliseconds. In general, it's best if round-trip times are under 200 milliseconds. The time it takes a packet to reach its destination is called latency. If you see a large variance in the round-trip times (which is called "jitter"), you are going to see poor performance talking to the host. However, a couple of laggards in a large sample (50 to 100) is no cause for worry.

To stop ping, type control-c. This terminates the program and prints out a nice summary of the number of packets transmitted, the number received, and the percentage of packets lost, plus the minimum, average, and maximum round-trip times of the packets.

As you can see, ping is a useful tool to test network connectivity and to measure whether packets are getting from one host to another and details about their journeys. If you're interested in a more sophisticated tool, let's talk about traceroute, which tells you the router path a packet is taking.

Traceroute: Cool Network Tool

While ping is useful, it only lets us measure how long a packet takes to get from one host to another. Wouldn't it be nice to have a tool that can trace out the path that a packet takes? Well, that's exactly what the appropriately named **traceroute** command does.

Traceroute on Unix and Linux (or `tracert` in the Microsoft world) attempts to trace the current network path to a destination. Here is an example of a traceroute run to `www.berkeley.edu`:

```
$ traceroute www.berkeley.edu
traceroute to amber.Berkeley.EDU (128.32.25.12), 30 hops max, 40 byte packets
1 sf1-e3.wired.net (206.221.193.1) 3.135 ms 3.021 ms 3.616 ms
2 sf0-e2s2.wired.net (205.227.206.33) 1.829 ms 3.886 ms 2.772 ms
3 paloalto-cr10.bbnplanet.net (131.119.26.105) 5.327 ms 4.597 ms 5.729 ms
4 paloalto-br1.bbnplanet.net (131.119.0.193) 4.842 ms 4.615 ms 3.425 ms
5 sl-sj-2.sprintlink.net (4.0.1.66) 7.488 ms 38.804 ms 7.708 ms
6 144.232.8.81 (144.232.8.81) 6.560 ms 6.631 ms 6.565 ms
7 144.232.4.97 (144.232.4.97) 7.638 ms 7.948 ms 8.129 ms
8 144.228.146.50 (144.228.146.50) 9.504 ms 12.684 ms 16.648 ms
9 f5-0.inr-666-eva.berkeley.edu (198.128.16.21) 9.762 ms 10.611 ms 10.403 ms
10 f0-0.inr-107-eva.Berkeley.EDU (128.32.2.1) 11.478 ms 10.868 ms 9.367 ms
11 f8-0.inr-100-eva.Berkeley.EDU (128.32.235.100) 10.738 ms 11.693 ms 11.520 ms
12 amber.Berkeley.EDU (128.32.25.12) 10.615 ms 10.693 ms 9.802 ms
```

Note how it lists the intermediary nodes (my people call 'em routers) between you and the destination. There are three sample times for each router that reflect how long the packet took to get from here to there.

To understand how to fully interpret the traceroute output, you need to know a bit about how it works. It uses two concepts I brought up in my earlier article about ping: the Time to Live (TTL) field in the IP packet (which tells us approximately how many router hops the packet can make before it dies or gets returned) and ICMP control messages (which are special IP control messages used to send network information between two hosts).

Traceroute works by addressing a packet to a (hopefully) unlistened-to UDP port on the destination machine (the default is port 33434). For the initial three packets, it sets the TTL to 1 and releases the packet. The packet then gets transferred to the first router (completing the first hop, in networkese), and the TTL gets decremented by the router from 1 to 0. The router then discards the packet and sends off an ICMP notification packet to our host with the message that the TTL expired from this router. This tells traceroute what the first hop is and how long it takes to get there (among other things). It repeats this, gradually incrementing the TTL until a path to the remote

host is traced and it gets back an ICMP Port Unreachable message, indicating that the remote host has been reached (that's why an unlistened-to port is used, so that the packet gets responded to instead of eaten by some random service).

Let's Go to Finland

Here is a more complicated example, trying to traceroute a host in Finland:

```
% traceroute www.hut.fi
traceroute to info-e.hut.fi (130.233.224.28), 30 hops max, 40-byte packets
1 derby-st.wired.net (206.221.201.241) 6 ms 5 ms 6 ms
2 206.221.201.129 (206.221.201.129) 33 ms 33 ms 33 ms
3 205.227.206.21 (205.227.206.21) 39 ms 36 ms 38 ms
4 sf0-e2s2.wired.net (205.227.206.33) 39 ms 50 ms 37 ms
5 131.119.26.105 (131.119.26.105) 38 ms 46 ms 39 ms
6 * paloalto-br1.bbnplanet.net (131.119.0.193) 43 ms 41 ms
7 vienna1-br1.bbnplanet.net (4.0.2.62) 164 ms 161 ms (ttl=248!) 127 ms
8 washdc1-br1.bbnplanet.net (4.0.1.89) 118 ms (ttl=246!) 122 ms (ttl=246!) 103 ms (ttl=246!)
9 192.157.65.121 (192.157.65.121) 101 ms (ttl=245!) 102 ms (ttl=245!) 101 ms (ttl=245!)
10 icm-pen-1-H1/0-T3.icp.net (198.67.131.18) 107 ms (ttl=244!) 107 ms (ttl=244!) 115 ms (ttl=244!)
11 icm-pen-11-P4/0-OC3C.icp.net (198.67.142.74) 107 ms 110 ms 107 ms
12 icm-t12-10-P1/0-STM1.icp.net (198.67.142.78) 198 ms 240 ms 198 ms
13 198.67.142.126 (198.67.142.126) 280 ms 218 ms *
14 198.67.142.222 (198.67.142.222) 222 ms 230 ms 224 ms
15 * * *
16 fi-gw.nordu.net (192.36.148.54) 216 ms (ttl=241!) 224 ms (ttl=241!) 217 ms (ttl=241!)
17 hutnet-gw.csc.fi (128.214.248.65) 237 ms (ttl=238!) 220 ms (ttl=238!) 226 ms (ttl=238!)
18 hutnet-gw.hut.fi (193.166.43.253) 220 ms 237 ms 223 ms
19 info-e.hut.fi (130.233.224.28) 291 ms (ttl=46!) 227 ms (ttl=46!) 267 ms (ttl=46!)
```

OK, so it's a bit long. The first line just summarizes what traceroute is going to do: trace the route to `www.hut.fi` (which is actually called `info-e.hut.fi`) for a maximum of 30 hops, using 40-byte packets.

The trip then goes through 19 routers, or 19 hops. The first one, which takes 5 to 6 milliseconds, is to my router at home and the second hops across my ISDN line, which takes 30 or so milliseconds. It then winds its way through our corporate LAN to BBNplanet. The seventh and twelfth hops are interesting, 'cause that's where times increase dramatically as it is going across the country and then across the Atlantic Ocean.

Sometimes times increase dramatically because the packet is crossing long distances, and other times the increases come from network congestion. You should be suspicious of a radical increase in one or two of the probes. If this happens, you can check things out by pinging the router a couple of times to see if packets are being dropped or if there is a great variance in times.

Also, starting at the seventh hop, we start seeing `(ttl=`*number*`!)` next to the times. This is an indication from traceroute that the TTLs it sees coming back don't match what it sent. This is an indication that

there may be an asymmetric path. An asymmetric path is one in which a packet takes one route going in one direction and a different route coming back. This is common nowadays.

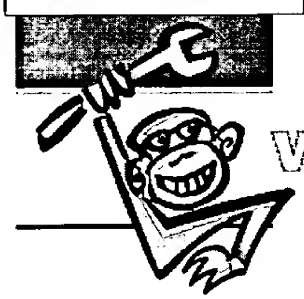
Note the asterisks on lines 6, 13, and 15. They indicate that a response wasn't received. Because there was no response from router 15, an educated guess would be that that router doesn't issue TTL-expired ICMP messages. The asterisks on line 6 and 13 are probably the result of dropped packets.

Combined with ping, traceroute makes a good network troubleshooting tool. By looking at traceroute output and looking for hops that have excessive times or dropped packets, one can find potential trouble spots in the path. Then by pinging the suspected hop, plus the hop before it, a larger sample of data can be accrued to determine if there is a problem.

A note of caution: Traceroute was designed for network testing. It should be used manually. Using it from scripts could adversely affect network performance.

Coda: Traceroute was written by Van Jacobson. Van heads up the Network Research Group at Lawrence Berkeley National Labs. His group has contributed greatly to the state of the TCP/IP protocols we use today. One of the cool things that NRG is currently working on is a program called **pathchar**, a tool that infers the characteristics of Internet paths. It infers the available bandwidth and size of the pipe per hop. Currently pathchar is in testing and only available for select Unix and Linux platforms. Test copies of the pathchar binary are available at ftp://ftp.ee.lbl.gov/pathchar. If you have a Unix or Linux platform to run it on, I'd recommend checking it out.

Rob Robertson is a senior network engineer at Wired Digital. He thinks he's going to get a Webmonkey messenger bag if he writes one more article. A ha ha ha. Heh.



Networks

Pages:
1 **Examine Your Network with Ping and Traceroute**
2 Traceroute: Cool Network Tool
3 Let's Go to Finland

Examine Your Network with Ping and Traceroute

by Rob Robertson *23 Oct 1997*

Rob Robertson is a senior network engineer at Wired Digital. He thinks he's going to get a Webmonkey messenger bag if he writes one more article. A ha ha ha. Heh.

Page 1

Web pages can be slow for any number of reasons, but it usually comes down to this: either the network is slow and congested, or the actual server you're accessing is heavily loaded and taking a while to process your request.

It is hard to diagnose a server problem remotely, but some basic tools can help you check out the network. The three tools I use most often to debug networks are called **ping**, **traceroute**, and **telnet**. All three tools originated under Unix, but have spawned DOS and Windows programs that behave similarly (namely ping and **tracert**, which are available using the DOS command shell). There are also versions of these programs that work on a Macintosh. For the purposes of this article, I'm assuming you are using Unix or Linux, though it should apply directly to DOS and Windows.

In this article I am going start with ping, then show how to use Traceroute. What ping does is send out a special packet called the **Internet Control Message Protocol (ICMP) Echo Request packet**. ICMP packets are special IP control messages that are used to send network information between two hosts, usually things like "don't do that," "send fewer packets," "we don't provide what you want," and "don't go here - go there." When a machine receives an Echo Request, it responds with an Echo Reply, placing the original Echo Request packet into the data field of the Echo Reply.

The way you use ping is pretty simple: You enter `ping hostname`, where `hostname` can be a machine name or just an IP address. There are many different versions of ping. The really lame ones just print out, "hostname is alive" (which is only cool if you are pinging a host named elvis). If you have a good version of ping installed, it will produce output like:

```
$ ping www.webmonkey.com

PING webmonkey.com (198.168.37.209): 56 data bytes

64 bytes from 198.168.37.209: icmp_seq=0 ttl=253 time=0.398 ms

64 bytes from 198.168.37.209: icmp_seq=1 ttl=253 time=0.552 ms

64 bytes from 198.168.37.209: icmp_seq=2 ttl=253 time=0.554 ms

64 bytes from 198.168.37.209: icmp_seq=3 ttl=253 time=0.553 ms

64 bytes from 198.168.37.209: icmp_seq=4 ttl=253 time=0.554 ms

64 bytes from 198.168.37.209: icmp_seq=5 ttl=253 time=0.551 ms

64 bytes from 198.168.37.209: icmp_seq=6 ttl=253 time=0.552 ms

64 bytes from 198.168.37.209: icmp_seq=7 ttl=253 time=0.554 ms

64 bytes from 198.168.37.209: icmp_seq=8 ttl=253 time=0.554 ms

64 bytes from 198.168.37.209: icmp_seq=9 ttl=253 time=0.553 ms

^C

        ----localhost PING Statistics----

        10 packets transmitted, 10 packets received, 0% packet loss

        round-trip min/avg/max = 0.398/0.537/0.554 ms   $
```

What's happening is that I'm pinging `www.webmonkey.com` - that is, sending one ICMP Echo Request packet, every second, to `www.webmonkey.com`. When the ping program gets an Echo Reply back from the remote host (`www.webmonkey.com`), it prints out the response, giving me several interesting pieces of information: the first is the IP address of where it came from (usually this should be the IP address of `www.webmonkey.com`); the second is the sequence number, starting at 0; the third is the Time To Live field; and the last is the number of milliseconds it took to get a reply.

The sequence number indicates which ping packet it got a reply to. A skipped sequence number indicates a dropped packet, meaning either the Echo Request or the responding Echo Reply somehow got lost in the network. If this is low (low is under 1 percent of the time), there is nothing to worry about. If it is much higher, the network link has some problems. It's not unheard of to get occasional out-of-order responses or multiple replies, but if you get a lot of them, there's trouble.

The Time To Live (TTL) field can be interesting. Every IP packet that gets sent out has a TTL field which is set to a relatively

high number (in this case, ping packets get a TTL of 255). As the packet traverses the network, the TTL field gets decreased by one by each router it goes through; when the TTL drops to 0, the packet is discarded by the router. The IP spec says that the TTL should be set to 60 (though it's 255 for ping packets). The main purpose of this is so that a packet doesn't live forever on the network and will eventually die when it is deemed "lost." But for us, it provides additional information. We can use the TTL to determine approximately how many router hops the packet has gone through. In this case it's 255 minus *N* hops, where *N* is the TTL of the returning Echo Replies. If the TTL field varies in successive pings, it could indicate that the successive reply packets are going via different routes, which isn't a great thing.

The time field is an indication of the round-trip time to get a packet to the remote host. The reply is measured in milliseconds. In general, it's best if round-trip times are under 200 milliseconds. The time it takes a packet to reach its destination is called latency. If you see a large variance in the round-trip times (which is called "jitter"), you are going to see poor performance talking to the host. However, a couple of laggards in a large sample (50 to 100) is no cause for worry.

To stop ping, type `control-c`. This terminates the program and prints out a nice summary of the number of packets transmitted, the number received, and the percentage of packets lost, plus the minimum, average, and maximum round-trip times of the packets.

As you can see, ping is a useful tool to test network connectivity and to measure whether packets are getting from one host to another and details about their journeys. If you're interested in a more sophisticated tool, let's talk about traceroute, which tells you the router path a packet is taking.

next page»